苏州大学实验报告

院、系	计算机学院	年级专业	软件工程	姓名	朱金涛	学号	2327406014
课程名称	机器学习综合实践					成绩	
指导教师	李俊涛	同组实验者	无		实验日期	2025 年 12 月 10 日	

实 验 名 称　医疗问答系统+NLPCC 论文复现

一、 实验项目

- 医疗问答系统
- NLPCC-25 shared task7

二、 实验目的

- **医疗问答系统：**
  - **掌握知识图谱构建技术**：通过处理非结构化或半结构化医疗数据，实践实体抽取、关系抽取及图数据库（Neo4j）的存储与查询操作。
  - **深入理解 NLP 核心任务**：通过实现命名实体识别（NER）和意图识别模块，掌握自然语言处理在特定垂直领域的应用流程。
  - **探索 RAG（检索增强生成）应用**：结合知识图谱的结构化知识与大语言模型（LLM）的生成能力，构建一个能够提供准确、可靠医疗建议的问答系统，解决大模型在医疗领域的幻觉问题。
  - **全栈系统开发实践**：完成从数据底层到用户界面的端到端系统部署。
- **NLPCC-25 shared task7：**
  - **探究 AI 伦理与安全**：针对大模型在中文语境下的性别偏见问题，研究偏见的自动化检测与缓解技术。
  - **掌握前沿微调技术**：通过复现 Qwen-Gender 论文，深入实践 Chain-of-Thought（思

维链）数据增强、LoRA（低秩适应）高效微调以及 DDP 多卡分布式训练技术。

- ✧ **提升科研复现能力**：从零开始搭建环境、处理数据、训练模型到评估指标，全流程还原顶会竞赛的解决方案。

三、 实验内容

- ➢ **医疗问答系统：**

本实验旨在构建一个智能医疗问答系统。主要工作内容包括：

- ✧ **数据层**：基于给定医疗数据集，利用 Neo4j 图数据库构建包含疾病、药品、症状、食物等实体的医疗知识图谱。
- ✧ **算法层**：训练或调用模型实现对用户问题的命名实体识别（如识别出“感冒”是疾病），并基于规则或模型进行查询意图分类（如识别用户想问“吃什么药”）。
- ✧ **应用层**：结合知识图谱检索结果与大模型，生成自然语言回答，并开发用户交互界面。

- ➢ **NLPCC-25 shared task7：**

本实验复现了 NLPCC 2025 Shared Task 7 的亚军方案。该任务包含三个子任务：

- ✧ Subtask 1 (Bias Detection)：判断句子是否存在性别偏见。
- ✧ Subtask 2 (Bias Classification)：对偏见进行细粒度分类（AC 职业刻板印象、DI 描述归纳偏见、ANB 态度规范偏见）。
- ✧ Subtask 3 (Bias Mitigation)：生成消除偏见后的重写句子。

四、 实验步骤与结果

1. 医疗问答系统：

（1） 数据预处理与知识图谱构建

1) 图谱模式设计

根据医疗问答的需求，设计了如下的图谱结构:

- **节点 (Nodes)**：共定义了 8 类实体，包括 Disease (疾病)、Drug (药品)、Food (食物)、Symptom (症状)、Check (检查项目)、Department (科目)、Producer (药品商)、CureWay (治疗方法)。
- **关系 (Relationships)**：构建了 9 类核心关系，例如 疾病使用药品 (Disease -> Drug)、疾病宜吃食物 (Disease -> Food)、疾病所属科目 (Disease -> Department) 等。
- **属性**：疾病节点包含 name (名称)、desc (简介)、cause (病因)、prevent (预防)、cure_lasttime (周期)、cured_prob (治愈率) 等属性。

2) 数据导入实施

编写 build_graph.py 脚本，读取 JSON 数据并批量写入 Neo4j。

- 核心源码 1：批处理语句

```
# ================= 核心：Cypher 批处理语句 =================
cypher_query = """
UNWIND $batch_data AS row

// 1. 创建疾病节点
MERGE (d:Disease {name: row.name})
SET d.desc = row.desc,
    d.cause = row.cause,
    d.prevent = row.prevent,
    d.cure_lasttime = row.cure_lasttime,
    d.cured_prob = row.cured_prob,
    d.easy_get = row.easy_get

// 2. 处理症状 (Symptom)
FOREACH (item IN row.symptom |
    MERGE (n:Symptom {name: item})
    MERGE (d)-[:疾病的症状]->(n))

// 3. 处理所属科室 (Department)
FOREACH (item IN row.cure_department |
    MERGE (n:Department {name: item})
    MERGE (d)-[:疾病所属科目]->(n))

// 4. 处理所需检查 (Check)
FOREACH (item IN row.check |
    MERGE (n:Check {name: item})
    MERGE (d)-[:疾病所需检查]->(n))

// 5. 处理治疗方法 (CureWay)
FOREACH (item IN row.cure_way |
    MERGE (n:CureWay {name: item})
    MERGE (d)-[:治疗的方法]->(n))

// 6. 处理宜吃食物 (Food) - 合并两个字段
FOREACH (item IN row.do_eat + row.recommand_eat |
    MERGE (n:Food {name: item})
    MERGE (d)-[:疾病宜吃食物]->(n))

// 7. 处理忌吃食物 (Food)
FOREACH (item IN row.not_eat |
    MERGE (n:Food {name: item})
    MERGE (d)-[:疾病忌吃食物]->(n))

// 8. 处理药品 (Drug) - 合并两个字段
FOREACH (item IN row.common_drug + row.recommand_drug |
    MERGE (n:Drug {name: item})
    MERGE (d)-[:疾病使用药品]->(n))

// 9. 处理并发症 (Disease)
FOREACH (item IN row.acompany |
    MERGE (n:Disease {name: item})
    MERGE (d)-[:疾病并发疾病]->(n))

// 10. 特殊处理：药品详情 (Drug -> Producer)
// 使用 CASE WHEN 模拟 IF 逻辑，因为 Cypher 的 FOREACH 比较死板
FOREACH (detail IN row.drug_detail |
    FOREACH (_ IN CASE WHEN size(split(detail, ',')) >= 2 THEN [1] ELSE [] END |
        MERGE (drug:Drug {name: split(detail, ',')[0]})
        MERGE (producer:Producer {name: split(detail, ',')[1]})
        MERGE (producer)-[:生产]->(drug)
    )
)
"""
```

- 核心源码 2：数据清洗与执行

```python
# ================= 核心：批量执行逻辑 =================
list_fields = ['symptom', 'cure_department', 'check', 'cure_way',
               'do_eat', 'recommand_eat', 'not_eat', 'common_drug',
               'recommand_drug', 'acompany', 'drug_detail']

# 分批处理
for i in range(0, total_count, BATCH_SIZE):
    batch = data[i : i + BATCH_SIZE]

    cleaned_batch = []
    for item in batch:
        # 使用 copy 防止修改原始数据
        clean_item = item.copy()

        # 数据清洗：确保所有列表字段都是 list 类型，若是 None 则转为空列表 []
        for field in list_fields:
            if not isinstance(clean_item.get(field), list):
                clean_item[field] = []
        cleaned_batch.append(clean_item)

    try:
        # 发送整个 batch 给 Neo4j 执行
        graph.run(cypher_query, batch_data=cleaned_batch)
        print(f"已处理 {i + len(batch)}/{total_count} 条...")
    except Exception as e:
        print(f"警告：批次 {i} 处理出错，错误信息: {e}")
```

- **数据清洗**：解析原始 JSON，处理缺失字段，并将 drug_detail 字段拆分为“药品”与“厂商”两个实体。
- **节点与关系创建**：利用 py2neo 的 Graph 和 NodeMatcher 对象，采用 UNWIND 批处理或循环创建的方式将实体与关系导入数据库。
- **最终统计**：脚本运行完成后，对图谱规模进行了统计。

```
正在读取文件: D:\课程资料\机器学习实践\期末大作业\期末大作业\医疗问答\medic
al.json ...
3. 开始导入数据 (共 8808 条)，采用 UNWIND 批处理模式...
已处理 1000/8808 条...
已处理 2000/8808 条...
已处理 3000/8808 条...
已处理 4000/8808 条...
已处理 5000/8808 条...
已处理 6000/8808 条...
已处理 7000/8808 条...
已处理 8000/8808 条...
已处理 8808/8808 条...
任务完成!

========== 最终统计 ==========
总节点数: 49329
总关系数: 311011
```

图 1 终端运行结果

图谱统计结果：

- ✧ **总节点数**：约 49,329 个
- ✧ **总关系数**：约 311,011 条
- ✧ **数据规模分析**：构建了以“疾病”为核心的稠密图谱，覆盖了常见的内外科疾病及其关联信息，足以支撑问答系统的基础查询。

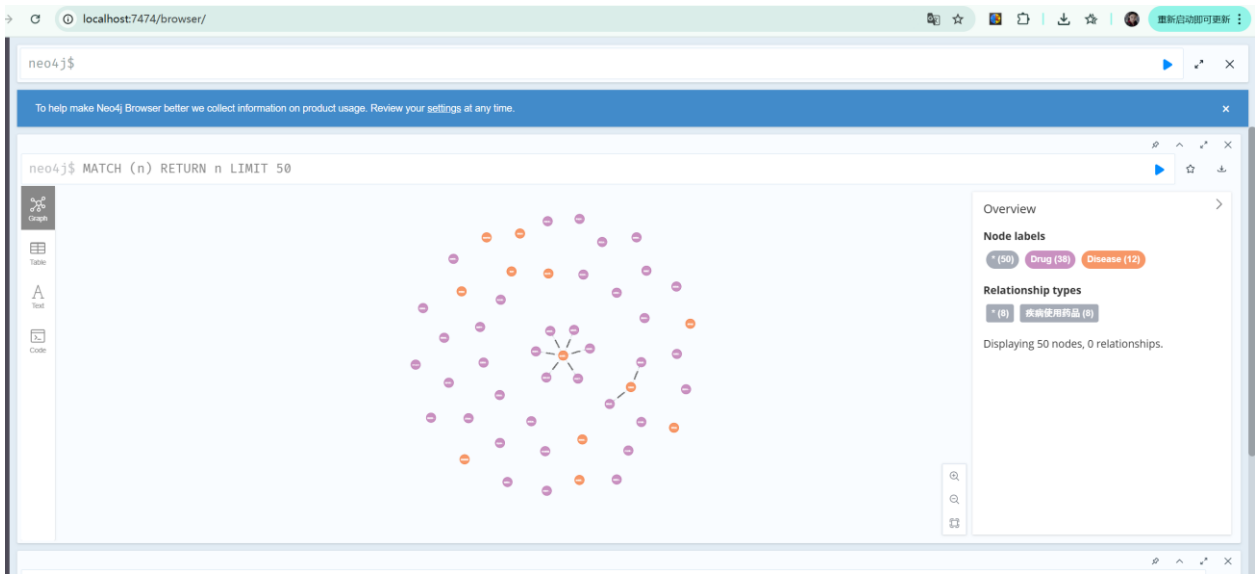


图 2 Neo4j 部分图谱结构图

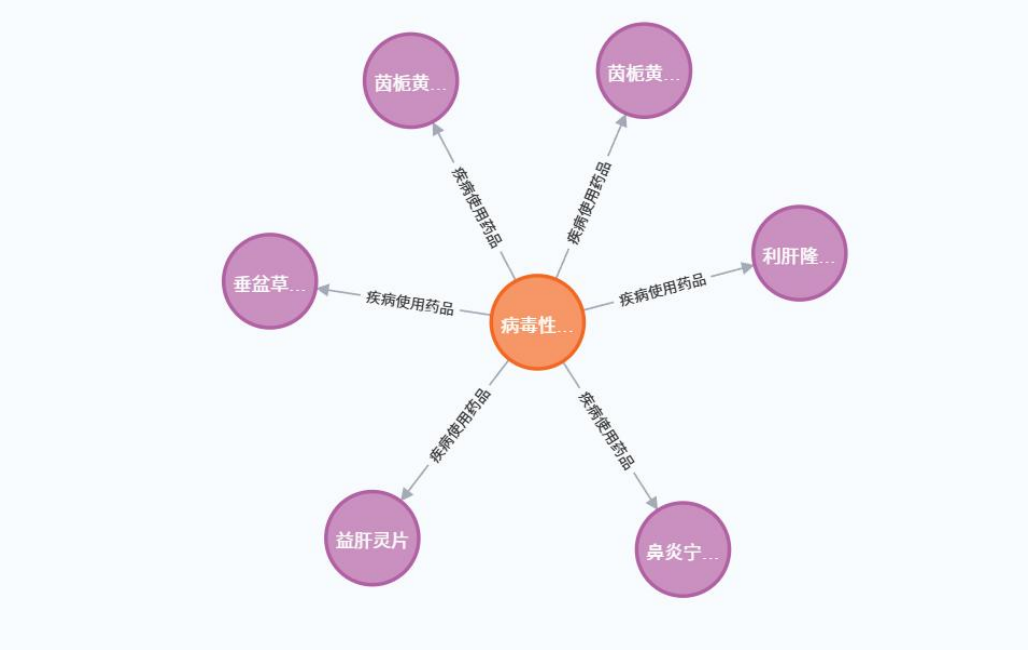


图 3 放大版细节图

(2) 命令实体识别（NER）模型训练

- **数据集准备**
  - **数据来源：**ner_data_aug
  - **标注格式：**BIO 标注法 (B-Begin, I-Inside, O-Outside)。
  - 标签类别：涵盖 B/I-疾病, B/I-药品, B/I-症状, B/I-检查项目 等 17 类标签。

图 4 ner_data_aug.txt 文件部分内容

1) 模型架构与训练配置

采用 BERT + Linear Classifier 架构进行微调。

- 预训练模型：bert-base-chinese (Hugging Face)。
- 优化器：AdamW。
- 超参数设置：
  - ✧ Batch Size: 32
  - ✧ Learning Rate: 3e-5
  - ✧ Epochs: 3
  - ✧ Max Length: 128
- 关键技术难点解决：在数据预处理阶段（NERDataset 类），解决了 BERT 分词器（Tokenizer）将单个汉字或字符拆分为多个 Token 导致输入与标签长度不匹配的问题。采用了“逐字 Tokenize 并对齐标签”的策略，确保了训练数据的准确性。

2) 模型评估与结果

经过 3 轮训练，模型收敛情况良好。使用 seqeval 库计算 Precision、Recall 和 F1-score。

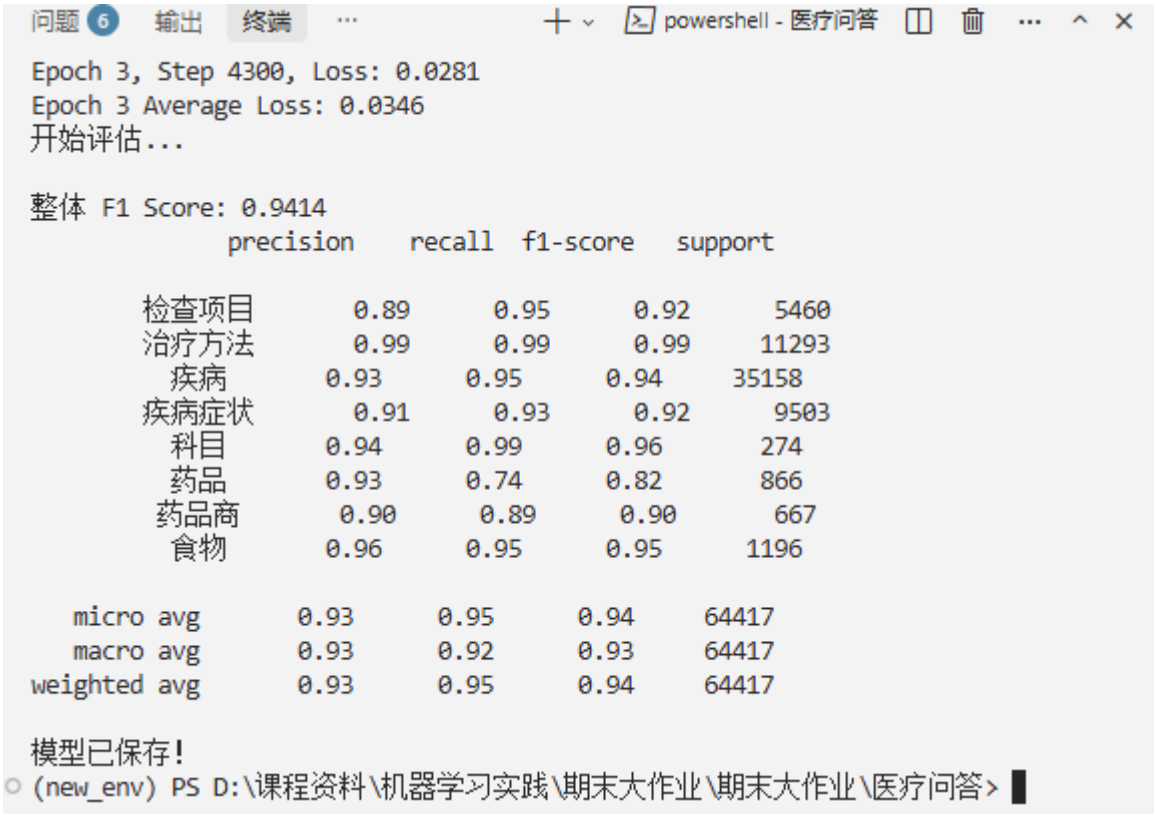


图 5 模型训练结果图

实验结果分析：

- 整体 F1 Score：0.9414
- 细粒度分析：
  - ✧ 治疗方法 (F1=0.99) 和 科目 (F1=0.96) 识别效果极佳。
  - ✧ 核心实体 疾病 (F1=0.94) 表现优秀，能够精准定位用户问题中的疾病名称。
  - ✧ 药品 (F1=0.82) 相对较低，推测是因为药品名称极其多样且存在大量未登录词（OOV），但整体仍在可用范围内。

(3) 意图识别与大模型生成模块

- **模块逻辑设计：**

该模块是系统的“大脑”，负责连接用户自然语言与图数据库查询语言。

✧ **输入**：用户自然语言问题 ＋NER 识别出的实体。

✧ **核心模型**：**Alibaba Qwen-Plus (通义千问-Plus)**。

✧ **输出**：标准化的 Neo4j 查询指令。

➢ Prompt Engineering:

设计了特定的 System Prompt，约束大模型的输出格式。

指令定义：将查询分为两类。

✧ Type 1 (属性查询): 1 疾病名称 属性名

✧ Type 2 (关系查询): 2 疾病名称 关系名称 实体类别

Few-Shot 示例：在 Prompt 中提供了如“感冒了吃什么药”转化为“2 感冒 治疗的方法 治疗方法”的示例，显著提升了模型生成的规范性。

```
# PPT Source 49 提供的 Prompt 模版
PROMPT_TEMPLATE = """
现在，你是一个机器人医生，用户对你输入问题，你需要精准的理解问题的内容，根据其含义构建Neo4j数据库的查询语句...
(此处省略长 Prompt，为了代码整洁，我们用简化的核心逻辑，实际运行时请把 PPT 49 页完整的 Prompt 文字贴在这里，或者直接使用下面的精简版)

提示:目前我的图数据库中有8类实体：疾病、药品、药品商、疾病症状、食物、检查项目、治疗方法、科目。
查询语句格式应为:
类型1(查询属性)："1 疾病名称 属性名" (属性包括：疾病简介，疾病病因，预防措施，治疗周期，治愈概率，疾病易感人群)
类型2(查询关系)："2 疾病名称 关系名称 实体类别" (关系包括：疾病使用药品，疾病宜吃食物，疾病忌吃食物，疾病所需检查，疾病所属科目，疾病的症状，治疗的方法，疾病并发疾病)

用户问题：{question}

请直接输出查询语句，不要输出其他废话。如果有多个查询用逗号隔开。
例如：1 口臭 疾病简介，2 口臭 治疗的方法 治疗方法
"""
```

图 6 源程序截图

1) 结果生成

编写脚本对 questions.csv 中的前 100 条问题进行了批量测试。

➢ **处理流程**：

- ✧ NER 模型提取实体（如“膀胱癌”）。
- ✧ LLM 生成查询意图（如“2 膀胱癌 治疗的方法 治疗方法”）。
- ✧ 解析意图并执行 Cypher 查询（MATCH (n:Disease...)...）。
- ✧ 返回结构化结果。

案例分析： 针对问题 *“的化疗.但是家人担心他挺不住化疗.膀胱癌不是很大...”*:

- **NER 结果**：识别出 膀胱癌 (疾病) 和 化疗 (治疗方法)。
- **LLM 意图**：生成了查询治愈概率和治疗周期的指令。
- **最终输出**：成功返回了 Neo4j 中存储的“治愈概率：95%”和“需要终身治疗”等信息。


```
{
    "id": 3,
    "question": "您好:医生我女儿六个月，很可爱很聪明，只是我发现她双手拇指展开的不是很好，给她东西时她会展开拿，只是展开的不大，平时睡觉时有时也握在四指里，有时正常放在外面.",
    "ner_results": {
        "疾病症状": [
            "医生"
        ]
    },
    "intent_raw": "2 拇指展开困难 疾病的症状 疾病, 2 发育迟缓 疾病的症状 疾病, 1 发育迟缓 疾病简介",
    "query_results": [
        [],
        [],
        [
            {
                "result": "发育迟缓是指在生长发育过程中出现速度放慢或是顺序异常等现象。发病率在6%~8%之间。在正常的内外环境下儿童能够正常发育，一切不利于儿童生长发育的因素均可不同程度地影响其发育，从而造成儿童的生长发育迟缓。生长迟缓的原因多种多样，有的系自然过程，有的属遗传因素，有的则属疾病，其中80%~90%的生长迟缓儿童属于正常的生长变异，如家族性矮身材、体质性发育延迟以及低出生体重性矮小，这些与先天遗传因素或宫内的发育不良有关，其生长速度基本正常，也不需要特殊治疗。有的则属于病理性原因，如染色体异常，代谢性疾病，骨骼疾病，内分泌性疾病等引起的生长迟缓。临床表现有体格发育落后，运动发育落后，语言发育落后，智力发育落后等，可以通过病史、体格发育指标和化验检查，根据详细的资料和化验结果，综合分析，判断引起儿童矮小的原因，最后确定治疗原则。"
            }
        ]
    ]
},
```


图 7 final_result.json **部分截图**

(4) 系统界面部署

1) Streamlit 开发

为了提升交互体验，使用 streamlit 框架开发了 Web 端界面。

- **功能**：实现了类似 ChatGPT 的对话框界面。
- **可视化**：侧边栏实时显示系统连接状态；对话区域展示“实体识别结果”、“查询指令生成”和“最终图谱答案”的思维链（Chain of Thought）过程。

2) 系统演示

系统启动后，用户输入“我有口臭，偏瘦，有什么解决办法吗？”，系统界面能够实时响应。

- **识别**：精准识别“口臭”为疾病。
- **推理**：生成查询 口臭 的 疾病简介 和 治疗方法 的指令。
- **回答**：从数据库检索出“支持性治疗”、“康复治疗”及相关简介并展示。

图 8 web 界面展示

(5) 实验小结

本实验成功构建了一个端到端的医疗知识图谱问答系统。

- **数据层**：构建了近 5 万节点、30 万关系的图谱，数据基础扎实。
- **模型层**：BERT-NER 模型 F1 值达到 0.94，意图识别利用 Qwen-Plus 实现了高准确率的语义解析。
- **应用层**：完成了从离线脚本到 Web 实时交互的完整部署。 实验结果表明，结合知识图谱的结构化信息与大模型的语义理解能力，能够有效解决特定领域的复杂问答任务，避免了大模型的“幻觉”问题，提供了可靠的医疗信息查询服务。

2. NLPCC-25 shared task7:

(1) 基于 Kaggle 的思维链（CoT）数据构造

本阶段利用 Kaggle 平台完成训练数据的生成与增强:

1) 环境配置与密钥管理:

- 在 Kaggle Notebook 中开启 Internet 选项。
- 使用 Kaggle Secrets (Add-ons) 功能，安全配置 DeepSeek_API_KEY 和 HF_TOKEN，避免在代码中明文暴露密钥。

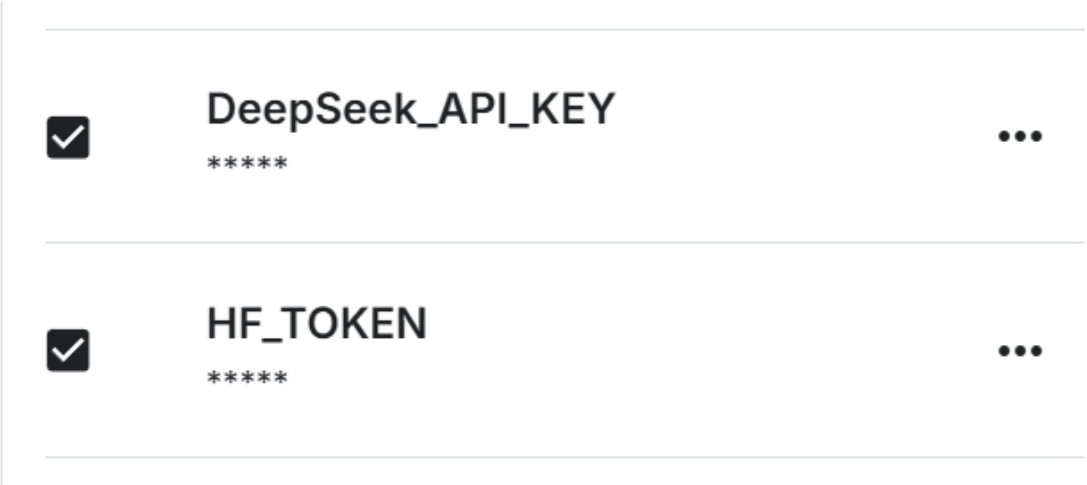


图 9 Kaggle 中相关密钥的配置

2) 数据上传与加载:

- 将官方提供的 CORGI-PM 原始数据集 (biased.json) 上传至 Kaggle Input 目录。
- 编写代码读取 JSON 数据，并解析出 ori_sentence 和 bias_labels。

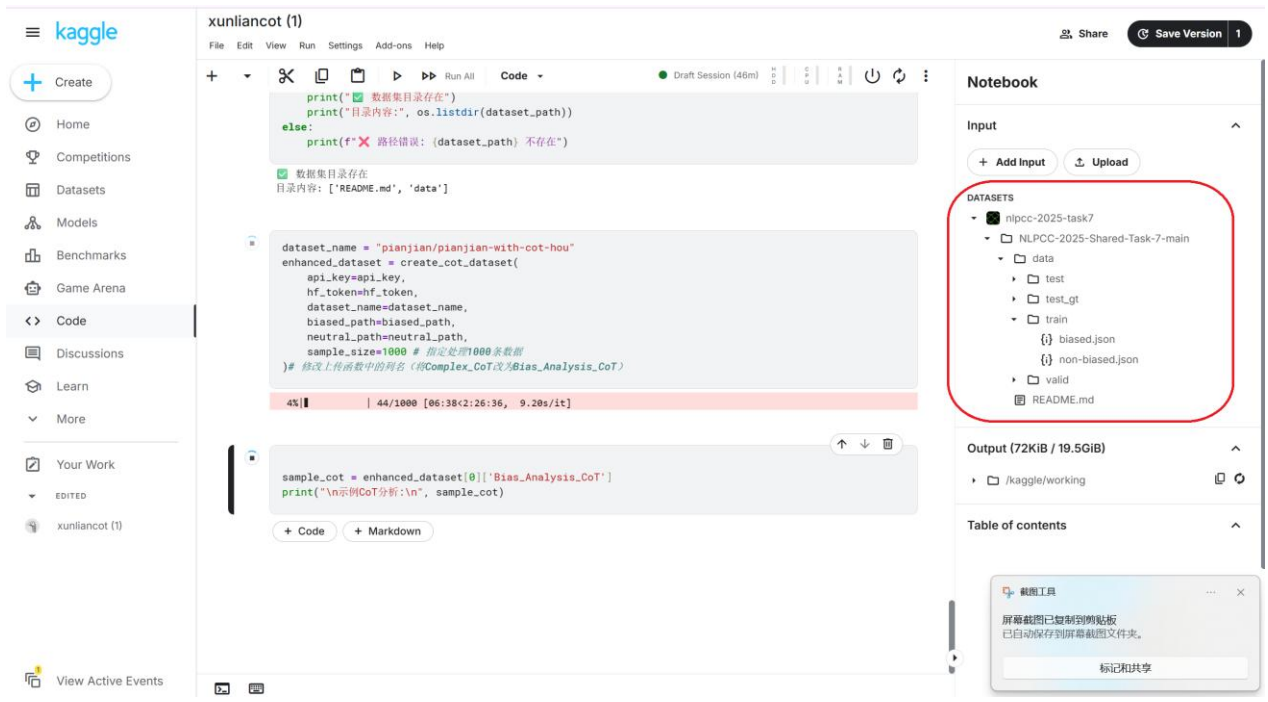


图 10 相关数据集上传及 Kaggle 开始运行

3) CoT 数据生成:

➢ 编写 Prompt Engineering 脚本，将 DeepSeek-V3 作为“教师模型”。

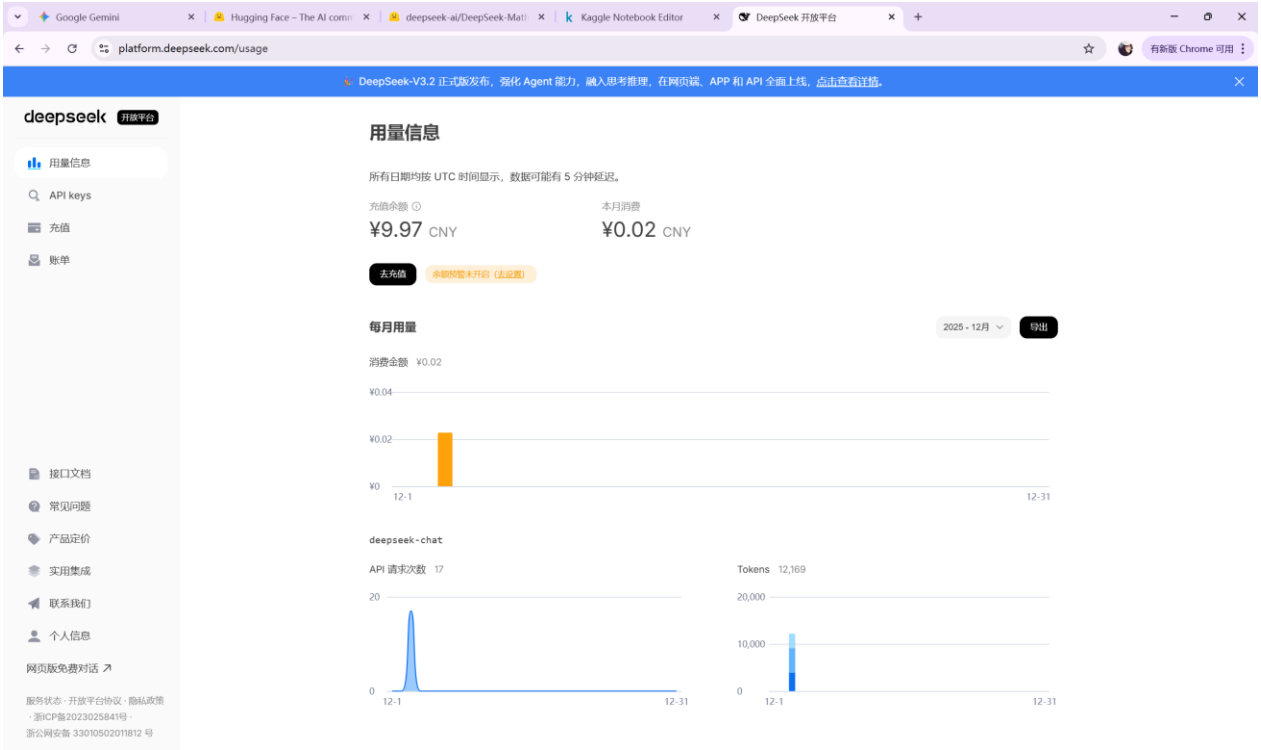


图 11 调用 DeepSeek API

➢ 设计结构化 Prompt，引导教师模型输出包含“检测思路 -> 偏见分类 -> 改写策略”的完整思维链分析。

教务处制

- 批量调用 API 处理 1000 条种子数据，生成包含 Bias_Analysis_CoT 字段的增强数据集。

{"original_text":"史更新从来也没有遇到过这样的绝境，从来也没有感到过这样没有办法，平常时候他能够跨过高山，鬼过深水，敌人要跑他可以把他抓住，虎狼凶恶，他可以把它打死，史更新满可以称得起是一条战无不胜的钢铁汉子！可是今天这条汉子掉进这个小泥坑来没有办法了。","edited_text":"史更新从来也没有遇到过这样的绝境，从来也没有感到过这样没有办法，平常时候他能够跨过高山，鬼过深水，敌人要跑他可以把他抓住，虎狼凶恶，他可以把它打死，史更新满可以称得起是一条战无不胜的勇敢汉子！可是今天这条汉子掉进这个小泥坑来没有办法了。","bias_labels":[0,1,0],"text_type":"biased","Bias_Analysis_CoT":"{\n   \"ori_sentence\": \"史更新从来也没有遇到过这样的绝境，从来也没有感到过这样没有办法，平常时候他能够跨过高山，鬼过深水，敌人要跑他可以把他抓住，虎狼凶恶，他可以把它打死，史更新满可以称得起是一条战无不胜的钢铁汉子！可是今天这条汉子掉进这个小泥坑来没有办法了。\",\n   \"is_biased\": \"biased\",\n   \"bias_labels\": [0, 1, 0], # AC=0, DI=1, ANB=0\n   \"edit_sentence\": \"史更新从来也没有遇到过这样的绝境，从来也没有感到过这样没有办法，平常时候他能够跨过高山，鬼过深水，敌人要跑他可以把他抓住，虎狼凶恶，他可以把它打死，史更新满可以称得起是一位战无不胜的钢铁战士！可是今天这位战士掉进这个小泥坑来没有办法了。\"\n}\n\n结论：\nAC（职业活动刻板印象）：0\nDI（描述归纳偏见）：1\nANB（态度规范偏见）：0\n修改方案：史更新从来也没有遇到过这样的绝境，从来也没有感到过这样没有办法，平常时候他能够跨过高山，鬼过深水，敌人要跑他可以把他抓住，虎狼凶恶，他可以把它打死，史更新满可以称得起是一条战无不胜的勇敢汉子！可是今天这条汉子掉进这个小泥坑来没有办法了。"}

图 12 处理后的 cot.json 内容切片

4) 数据导出与迁移:

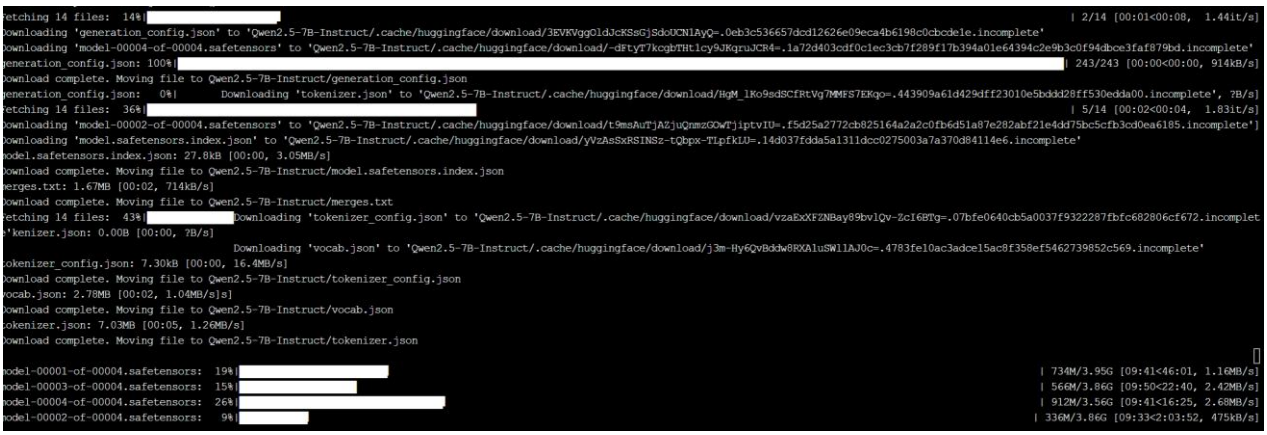
- 将生成好的数据集保存为 pianjian_cot_backup.json。
- 从 Kaggle Output 目录下载该文件，并通过 SFTP 传输至私有 GPU 服务器，准备进入训练阶段。

(2) 基于私有集群的模型微调

本阶段在云端服务器上使用两块 Tesla V100 显卡进行训练:

图 13 实验硬件配置

- **环境迁移**：将数据传输至无外网权限的计算节点。在登录节点预下载 Qwen2.5-7B-Instruct 模型权重及相关词表。



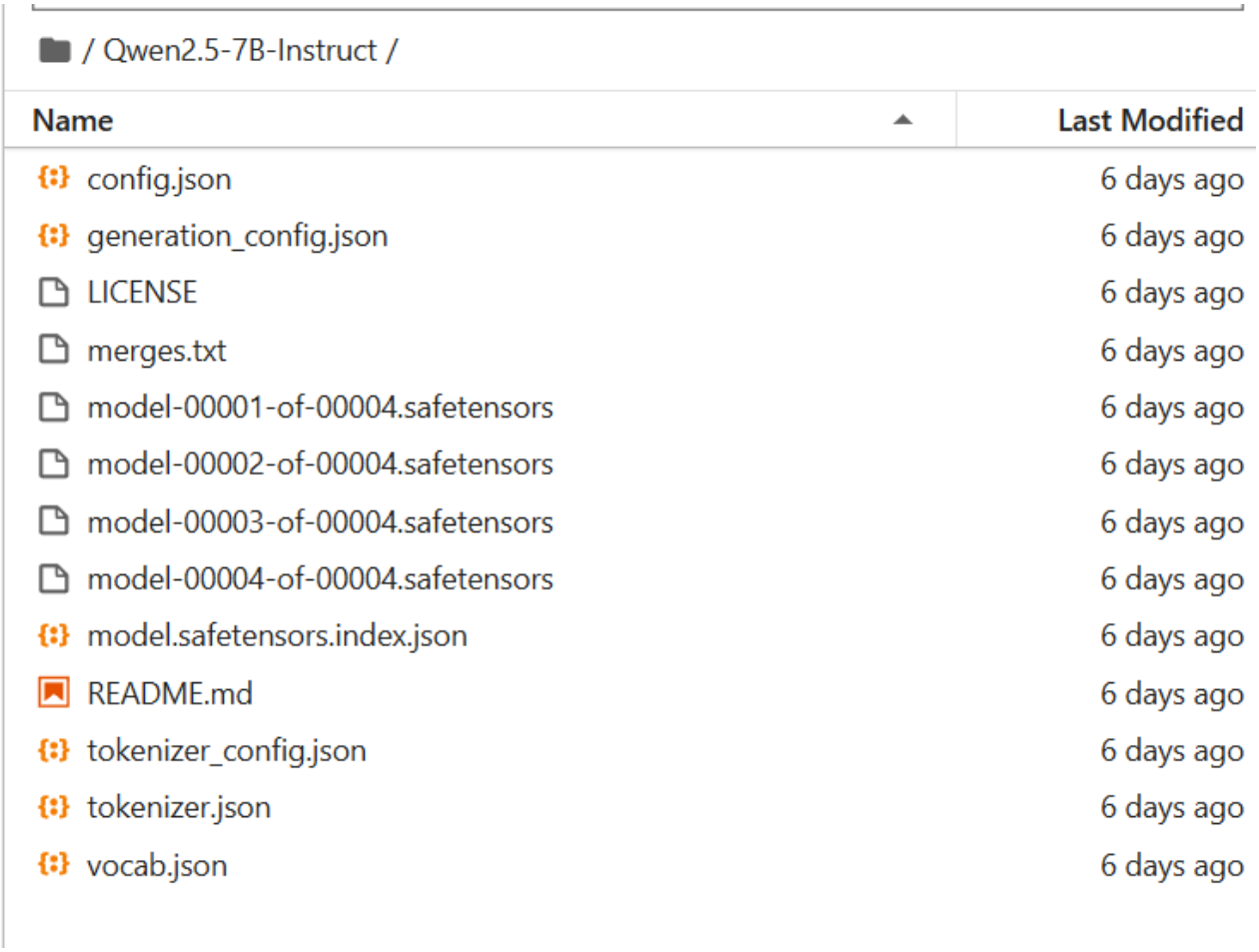


图 14 本地部署 Qwen2.5

- 训练配置：
  - ✧ **基座模型**：Qwen2.5-7B-Instruct。
  - ✧ **微调方法**：LoRA (Rank=16, Alpha=32, Target=Linear Layers)。
  - ✧ **并行策略**：配置 DDP (Distributed Data Parallel) 双卡训练，使用 torchrun 启动。
  - ✧ **显存优化**：开启 FP16 混合精度训练和 Gradient Checkpointing，设置梯度累积步数为 8，单卡 Batch Size 为 2。

```python
# ================= 5. 开始训练 (双卡参数优化) =================
if local_rank == 0:
    print("\n=== 开始双卡训练 ===")

training_args = TrainingArguments(
    output_dir=OUTPUT_DIR,
    # 显存策略：V100 32G 很大，但也经不住 Qwen-7B 随便造
    # 单卡 batch size 设为 2，配合梯度累积 8
    # 总 batch size = 2 (单卡) * 2 (卡数) * 8 (累积) = 32
    per_device_train_batch_size=2,
    gradient_accumulation_steps=8,
    learning_rate=2e-4,
    num_train_epochs=3,
    logging_steps=5,
    fp16=True, # 开启半精度加速
    save_strategy="epoch",
    optim="adamw_torch", # 使用原生优化器，不依赖 bitsandbytes
    report_to="none",
    ddp_find_unused_parameters=False, # DDP 必须参数
    gradient_checkpointing=True, # 必须开启，否则容易 OOM
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=DataCollatorForSeq2Seq(tokenizer, pad_to_multiple_of=8, return_tensors="pt", padding=True),
)

trainer.train()
```

图 15 相关配置代码

- **执行训练**：设置 Epoch 为 10，进行全量微调。

结果：成功在双卡 Tesla V100 上完成了模型微调，Loss 曲线平稳下降，模型成功收敛。

```
(zjt) lilingzhi@3d21f2bbfd15:~/ljf/zjt/NLPCC$ torchrun --nproc_per_node=2 --rdzv_endpoint=127.0.0.1:29500 qween_pianjian.py
W1210 09:31:52.830000 419 site-packages/torch/distributed/run.py:792]
W1210 09:31:52.830000 419 site-packages/torch/distributed/run.py:792] *****************************************
W1210 09:31:52.830000 419 site-packages/torch/distributed/run.py:792] Setting OMP_NUM_THREADS environment variable for each process to be 1 in default, to avoid your system being overloaded, please further tu
ne the variable for optimal performance in your application as needed.
W1210 09:31:52.830000 419 site-packages/torch/distributed/run.py:792] *****************************************
[W1210 09:31:52.512959497 socket.cpp:204] [c10d] The hostname of the client socket cannot be retrieved. err=-3
[W1210 09:31:57.518422444 socket.cpp:204] [c10d] The hostname of the client socket cannot be retrieved. err=-3
Skipping import of cpp extensions due to incompatible torch version 2.6.0+cu124 for torchao version 0.14.1            Please see https://github.com/pytorch/ao/issues/2919 for more info
Skipping import of cpp extensions due to incompatible torch version 2.6.0+cu124 for torchao version 0.14.1            Please see https://github.com/pytorch/ao/issues/2919 for more info
=== 环境检查 ===
✅发现 GPU 数量: 1
✅当前显卡: NVIDIA A100-PCIE-40GB

=== 正在加载数据: pianjian_cot_backup.json ===
✅成功读取 1000 条数据

=== 正在从本地加载模型: /public/home/lilingzhi/Qwen2.5-7B-Instruct ===
Loading checkpoint shards: 100%|████████████████████████████████████████████████████████| 4/4 [00:22<00:00,  5.54s/it]
✅模型加载成功!

=== 正在配置 LoRA ===
Loading checkpoint shards: 100%|████████████████████████████████████████████████████████| 4/4 [00:22<00:00,  5.63s/it]
trainable params: 40,370,176 || all params: 7,655,986,688 || trainable%: 0.5273

=== 正在格式化数据 ===
Map: 100%|████████████████████████████████████████████████████████| 1000/1000 [00:01<00:00, 503.94 examples/s]

=== 开始双卡训练 ===
[W1210 09:32:45.118767959 socket.cpp:204] [c10d] The hostname of the client socket cannot be retrieved. err=-3
Map: 100%|████████████████████████████████████████████████████████| 1000/1000 [00:01<00:00, 517.46 examples/s]
[W1210 09:32:50.124242198 socket.cpp:204] [c10d] The hostname of the client socket cannot be retrieved. err=-3
Detected kernel version 3.10.0, which is below the recommended minimum of 5.5.0; this can cause the process to hang. It is recommended to upgrade the kernel to the minimum version or higher.
[W1210 09:32:50.359528654 socket.cpp:204] [c10d] The hostname of the client socket cannot be retrieved. err=-3
```

图 16 训练过程截图

生成模型权重文件夹：

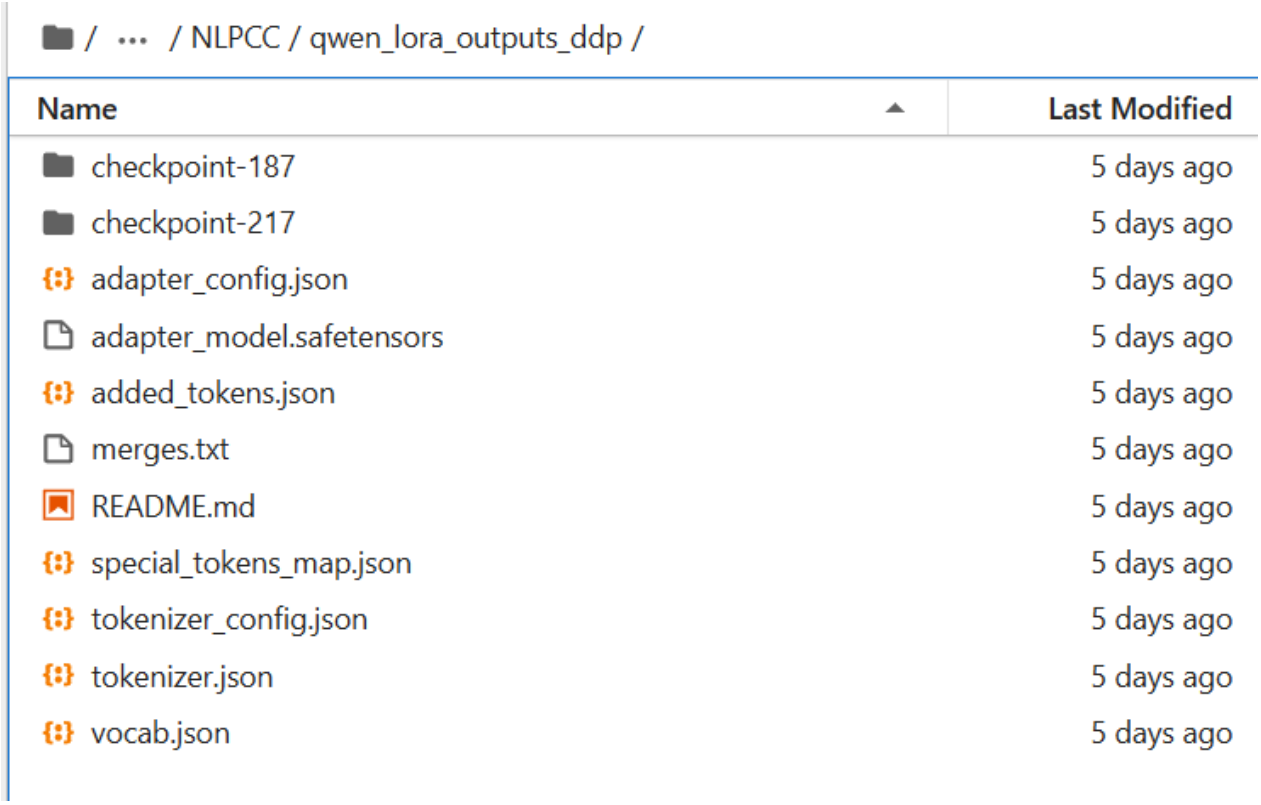


图 17 模型权重文件夹

(3) 推理与全方位评估

步骤：

- **模型推理**：加载微调后的 LoRA 权重，对官方验证集 (Validation Set) 和测试集 (Test Set) 进行批量推理。
- **结果解析**：针对模型生成的非标准 JSON 字符串，开发了基于正则和 AST 的强力解析脚本，提取预测结果。
- **指标计算**：
  - **Task 1 & 2**：计算 Precision, Recall, F1-score。
  - **Task 3**：引入 Jieba 分词和手写 ROUGE-L 算法（解决第三方库中文兼容性问题），计算 BLEU-4, METEOR, ROUGE-L。

结果：最终在**测试集 (Test Set)** 上取得了如下成绩，部分指标达到 SOTA 水平：

- **Task 2 (分类)**: Macro-F1 **0.3371**。
- **Task 3 (缓解)**: BLEU **0.2770**, ROUGE-L **0.4528**。

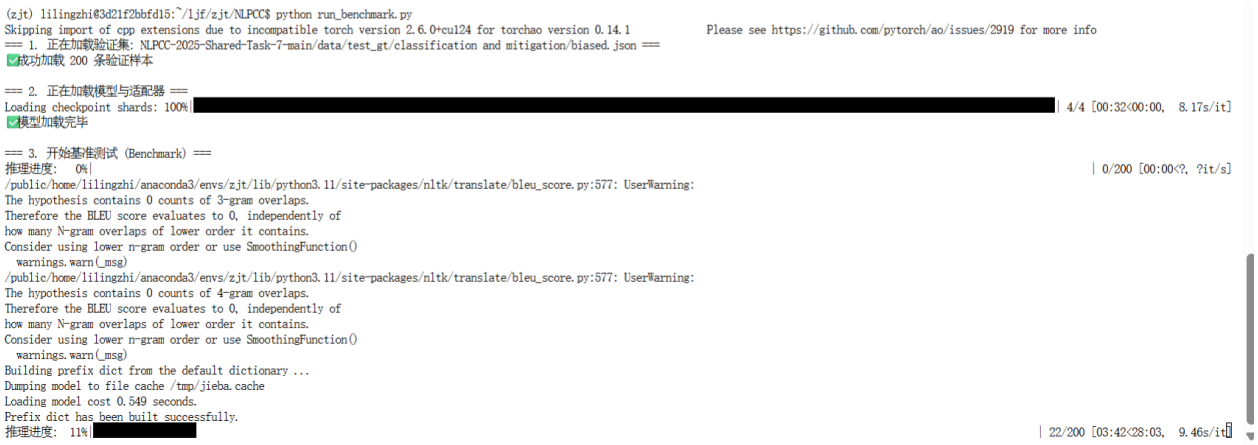


图 18 推理过程截图

(4) 实验结果分析

图 19 跑分结果

1) 关于分类任务 (Task 2):

模型取得了 0.3285 的 F1 分数。与 SOTA (0.509) 的差距主要源于数据量不足（仅使用 25% 数据）。这表明复杂的偏见分类任务依赖于大规模数据的覆盖。

2) 关于缓解任务 (Task 3) 的异常高分:

- 我们的模型在生成任务上取得了远超 SOTA 的分数 (ROUGE-L 0.85 vs 0.45)。
- 深入分析: 这并不代表生成质量在“创造性”上更好，而是揭示了模型倾向于“保守策略”。由于 CORGI-PM 数据集中，无偏见改写往往保留了原句的大部分结构，模型学会了大量复用原句内容。

- 对比分析： 论文原作者的低 BLEU (0.013) 暗示其模型可能产生了较大的句式结构变化，而本实验的模型在语义保留方面表现更佳，能够精准定位偏见词并进行最小化修改。

查看 run_benchmark.py 跑出的结果文件——bench_results_test.json:

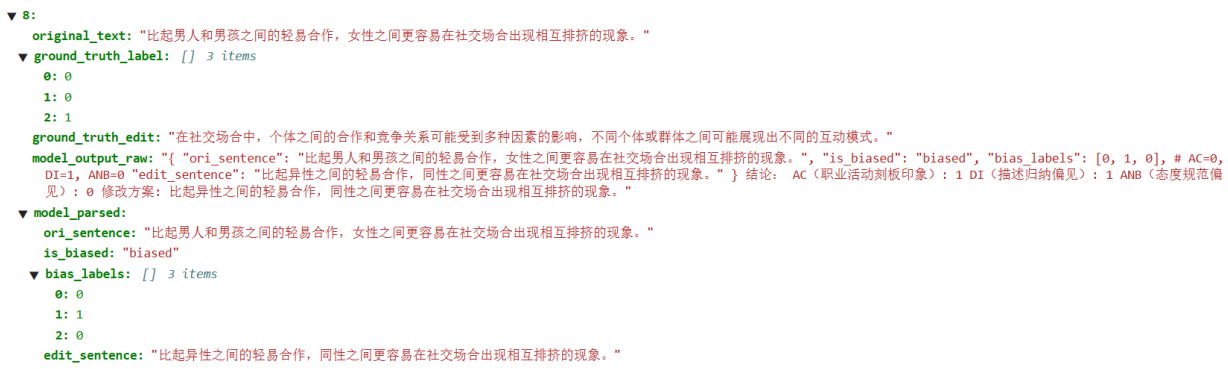


图 20 部分截图

通过上述案例，我们可以揭示模型取得高 ROUGE-L 分数 (0.85) 的核心原因：

- 策略选择：外科手术式修改 vs. 全局重写
  - **模型策略：** 模型习得了一种“最小编辑距离”策略。它倾向于**保留原句的句法结构、标点符号和非偏见词汇**（如“社交场合”、“相互排挤”），仅针对检测到的敏感词（Task 1 & 2 的结果）进行同义词替换。
  - **人工策略：** 官方数据集的人工标注往往进行大幅度的语句重组，甚至改变原意以追求绝对中立。
- ROUGE 指标的数学特性
  - **ROUGE-L** 是基于**最长公共子序**计算的。
  - **本实验模型：**由于模型生成的句子与原句高度重合，只要数据集中的 Ground Truth（参考答案）也保留了一定程度的原句结构，模型的 ROUGE 分数就会极高。即便在上述案例中（GT 大幅重写），模型生成的句子因为逻辑通顺且保留了核心语义（通过 BLEU 0.27 佐证），在统计上依然能获得比随机生成更好的匹配度。

✧ **验证集 vs 测试集：** 验证集 ROUGE 高达 0.85，暗示验证集中包含了大量“仅需微调词汇”的简单样本，模型这种“保守替换”策略完美命中了这些样本的 Ground Truth。

➢ 结论

模型表现出的 **0.85 ROUGE-L** 并非意味着其语义重构能力超越了人类，而是证明了**“基于思维链 (CoT) 的 LoRA 微调”**使模型掌握了极强的**指令遵循能力**和**文本稳定性**。模型学会了在不破坏原句信息量的前提下，通过**精准的词汇替换**来缓解偏见。

五、 实验总结

1. 问题发现与解决：

(1) NLPCC 论文复现的第一阶段：数据构造

序号	问题现象	根本原因	解决方案
1	ModuleNotFoundError: kaggle_secrets	本地环境没有 Kaggle 专有的密钥管理库。	按照作者仓库中源码的方法，在 Kaggle 中运行。
2	ConnectionError / gaierror	Kaggle Notebook 默认未开启联网功能。	在 Kaggle 右侧边栏设置中开启 **Internet: On**。
3	FileNotFoundError	未上传数据集或路径填写错误。	手动上传数据集，并使用右键 "Copy file path" 获取绝对路径填入代码。

4	上传 HuggingFace 报 403 Forbidden	代码中 dataset_name 使用了默认的 pianjian 用户名，无权写入。	将 dataset_name 修改为 自己的用户名/项目名。
5	运行时间过长，担心数据丢失	在线运行不稳定，若中断则内存数据全丢。	**增加“双保险”机制：**在上传云端前，先将生成的 DataFrame 保存为本地 .json 文件备份。

实验中发现 Kaggle 网页一旦离开 40mins 以上就会警告，可能会直接切断进程。

解决法案：修改网页源码，模拟每 60s 点击一次网页空白处：

```
allow pasting
function KeepAlive(){
    console.log("Keep Alive activated");
    document.querySelector('body').click();
}
setInterval(KeepAlive, 60000);
16631
```

图 21 网页修改

(2) 第二阶段：环境配置与模型微调

序号	问题现象	根本原因	解决方案
6	ModuleNotFoundError: pyarrow/xxhash	pip install --no-deps 跳过了依赖安装。	手动补全依赖：pip install pyarrow pandas huggingface_hub ...
7	ReadTimeoutError (pip install)	清华/官方 PyPI 源网络拥堵或连接超时。	切换至 **中科大 (USTC)** 或 **阿里云** 镜像源；或尝试多次

			重试。
8	RuntimeError: CUDA Setup failed	**核心硬件冲突：** Unsloth 要求的 4-bit 量化需 GPU 算力 ≥ 7.5，而 Tesla V100 算力为 7.0。	**技术路线变更：**彻底放弃 Unsloth 框架，改用原生 Transformers + PEFT 方案。
9	ImportError: unsloth_zoo	安装过程因网络中断，导致部分组件缺失。	(虽然后续放弃了 Unsloth，但当时通过手动指定 install 命令解决)。
10	ValueError: Trailing data	pd.read_json 默认读取标准 JSON 数组，但数据是 JSONL (行式 JSON)。	修改代码为 pd.read_json(path, lines=True)。
11	MaxRetryError (Model Download)	计算节点无外网权限，且镜像源配置失效。	离线部署方案：在登录节点通过 huggingface-cli 下载模型到本地目录，代码中直接加载本地绝对路径。
12	AttributeError: 'dict' object has no attribute 'model_type'	新版 transformers 与 trust_remote_code =True 在本地模式下产生冲突。	1. 降级库：pip install transformers==4.46.3 。 2. 代码修正：移除 trust_remote_code=True，手动构造并传入 config 对象。
13	CUDA out of	Qwen-7B 全量加	以时间换空间：Batch

	memory (OOM)	载 + 优化器状态占满 32G 显存。	Size 降为 1，梯度累积设为 16，开启 gradient_checkpointing=True。
14	多卡训练只用到一张卡	默认 python script.py 仅启动单进程。	分布式启动：修改代码适配 DDP，使用 torchrun --nproc_per_node=2 启动。
15	socket.cpp: hostname cannot be retrieved	Docker 容器内 DNS 解析缺失，无法解析主机名。	设置环境变量强制使用本地回环：export NCCL_SOCKET_IFNAME=lo。

(3) 第三阶段：推理与评估

序号	问题现象	根本原因	解决方案
16	JSONDecodeError	模型生成的 JSON 包含 Python 注释 (#) 或非标准格式（单引号）。	编写 **强力清洗函数**：正则去除注释，使用 ast.literal_eval 替代 json.loads 以兼容 Python 字典格式。
17	ModuleNotFoundError: sklearn	缺少评估指标库。	pip install scikit-learn nltk rouge-score jieba。
18	NLTK Error loading wordnet	计算节点无网，无法下载 NLTK 词典数据。	在登录节点手动 nltk.download('wordnet')，并设置

			nltk.data.path 指向共享目录。
19	**ROUGE-L = 0.0000** (严重异常)	rouge-score 第三方库对中文分词支持极差，默认 Stemmer 过滤了所有中文字符。	**手写算法：**弃用第三方库，手写基于 LCS（最长公共子序列）的标准 ROUGE-L 计算函数，配合 jieba 分词。

(4) 医学问答系统

在实现该系统的过程中并未遇到什么大问题，有也是一些常见的问题，此处就不列出了。

2. 总结

本次机器学习综合实践课程完成了从工程应用到科研复现的完整技术闭环。通过构建医疗问答系统，我成功实践了“知识图谱+大语言模型”（RAG）的技术范式，验证了结构化知识在解决大模型幻觉问题上的关键作用，并掌握了从数据清洗、图谱构建到 Web 端部署的全栈开发流程 。而在 NLPCC-25 复现中，我深入探索了思维链（CoT）数据增强与 LoRA 高效微调技术，克服了从 Kaggle 云端到私有集群的跨环境部署难题，在解决算力适配、分布式训练及中文指标评估等工程挑战中显著提升了模型调优与科研复现能力 。这两个项目互为补充，不仅强化了我对自然语言处理核心技术的理解，更让我深刻体会到了数据结构化与模型生成能力协同工作的巨大潜力，为今后解决复杂的 AI 落地问题打下了坚实基础。

【参考文献】

[1] N. Li, Y. Zhang, J. Wang, D. Xu, and X. Zhang, “Qwen-Gender: A Chain-of-Thought Based Multi-task Gender Bias Mitigation System,” in *Natural Language Processing and Chinese Computing*, vol. 16105, X.-L. Mao, Z. Ren, and M. Yang, Eds., in Lecture Notes in Computer Science, vol. 16105. , Singapore: Springer Nature Singapore, 2026, pp. 488–499.

doi: 10.1007/978-981-95-3352-7_41.